Pemodelan Visual dengan Menggunakan UML dan Rational Rose

Prastuti Sulistyorini
Program Studi Teknik Informatika
STMIK Widya Pratama Pekalongan

Abstrak : Saat ini piranti lunak semakin luas dan besar lingkupnya, sehingga tidak bisa lagi dibuat asal-asalan. Piranti lunak saat ini seharusnya dirancang dengan memperhatikan hal-hal seperti scalability, security, dan eksekusi yang robust walaupun dalam kondisi yang sulit. Selain itu arsitekturnya harus didefinisikan dengan jelas, agar bug mudah ditemukan dan diperbaiki, bahkan oleh orang lain selain programmer aslinya. Keuntungan lain dari perencanaan arsitektur yang matang adalah dimungkinkannya penggunaan kembali modul atau komponen untuk aplikasi piranti lunak lain membutuhkan fungsionalitas yang sama.
Dengan menggunakan model diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti scalability, robustnees, security, dan sebagainya. Untuk melakukan pemodelan sistem / perangkat lunak secara visual digunakan UML (Unified Modelling Language) yang digambarkan secara elektronik lewat sarana perangkat lunak Rational Rose.

Kata Kunci : UML, Rational Rose

PENDAHULUAN

Saat ini piranti lunak semakin luas dan besar lingkupnya, sehingga tidak bisa lagi dibuat asal-asalan. Piranti lunak saat ini seharusnya dirancang dengan memperhatikan hal-hal seperti scalability, security, dan eksekusi yang robust walaupun dalam kondisi yang sulit. Selain itu arsitekturnya harus didefinisikan dengan jelas, agar bug mudah ditemukan dan diperbaiki, bahkan oleh orang lain selain programmer aslinya. Keuntungan lain dari perencanaan arsitektur yang matang adalah dimungkinkannya penggunaan kembali modul atau komponen untuk aplikasi piranti lunak lain membutuhkan fungsionalitas yang sama.

Kebergantungan dunia bisnis terhadap sistem informasi berbasis komputer telah mempengaruhi peningkatan pasar piranti lunak, sekaligus juga menjadi tantangan bagi dunia rekayasa software untuk memiliki teknik rekayasa yang dapat meningkatkan kualitas serta mengurangi biaya dan waktu. Termasuk dalam teknik rekayasa software diantaranya adalah teknik pemodelan visual.

Pemodelan visual (*visual modeling*) merupakan proses menggambarkan cetak biru suatu sistem informasi secara grafis, terdiri dari komponan – komponen, interface, dan koneksi – koneksi yang ada dalam sistem tersebut, agar mudah dipahami dan dikomunikasikan. Visual modeling dapat membantu untuk menampilkan elemen – elemen yang penting secara detail dari suatu masalah yang kompleks dan menyaring untuk kemudian membuang elemen – elemen yang tidak penting. Membuat model dari sebuah sistem yang kompleks sangatlah penting karena tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin kompleks sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik.

Dengan menggunakan model diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti scalability, robustnees, security, dan sebagainya. Untuk melakukan pemodelan sistem / perangkat lunak secara visual digunakan UML (Unified Modelling Language) yang digambarkan secara elektronik lewat sarana perangkat lunak Rational Rose.

MENGENAL UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi,

merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti C++, Java, atau VB. NET.

DIAGRAM UML

Setiap sistem yang kompleks seharusnya bisa dipandang dari sudut yang berbeda – beda sehingga bisa mendapatkan pemahaman secara menyeluruh . Untuk upaya tersebut UML menyediakan 9 jenis diagram yang dapat dikelompokkan berdasarkan sifatnya statis atau dinamis. Ke 9 diagram dalam UML itu adalah :

1. Diagram Kelas

   Diagram kelas bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi serta relasi.

2. Diagram Objek

   Diagram objek bersifat statis. Diagram ini memperlihatkan objek-objek serta relasi antar objek. Diagram objek memperlihatkan instansiasi statis dari segala sesuatu yang dijumpai pada diagram kelas.

3. *Use case Diagram*

   Diagram ini bersifat statis. Diagram ini memperlihatkan himpunan use case dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.

4. *Sequence Diagram* (Diagram urutan)

   Diagram ini bersifat dinamis. Diagram sequence merupakan diagram interaksi yang menekankan pada pengiriman pesan (message) dalam suatu waktu tertentu.

5. *Collaboration Diagram*

   Diagram ini bersifat dinamis. Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi struktural dari objek – objek yang menerima serta mengirim pesan (message).

6. *Statechart Diagram*

   Diagram ini bersifat dinamis. Diagram ini memperlihatkan state – state pada sistem, memuat state, transisi, event, serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka, kelas, kolaborasi dan terutama penting pada pemodelan sistem – sistem yang reaktif.

7. *Activity Diagram*

   Diagram ini bersifat dinamis. Diagram ini adalah tipe khusus dari diagram state yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi – fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek.

8. *Component Diagram*

   Diagram ini bersifat statis. Diagram ini memperlihatkan organisasi serta kebergantungan pada komponen – komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas-kelas, antarmuka – antarmuka serta kolaborasi – kolaborasi.

9. *Deployment Diagram*

   Diagram ini bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (saat run time). Dengan ini memuat simpul – simpul (node) beserta komponen – komponen yang ada di dalamnya. Deployment diagram berhubungan erat dengan diagram kompoen dimana deployment diagram memuat satu atau lebih komponen – komponen. Diagram ini sangat berguna saat aplikasi berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*).

Ke 9 diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semua dibuat sesuai dengan kebutuhan.

MENGENAL RATIONAL ROSE

Rational Rose merupakan *tool* pemodelan secara visual untuk pengembangan sistem berbasis objek yang sangat handal untuk digunakan sebagai bantuan bagi para pengembang dalam melakukan analisis dan perancangan sistem. Rational Rose digunakan untuk melakukan pemodelan sistem sebelum pengembang menulis kode – kode dalam bahasa pemrograman tertentu. Rational Rose mendukung pemodelan bisnis, yang membantu para pengembang untuk memahami sistem secara komprehensif. Rational Rose juga membantu analisis sistem dengan cara pengembang membuat diagram use case untuk melihat fungsionalitas sistem secara keseluruhan sesuai dengan harapan dan keinginan pengguna. Rational Rose juga menuntut pengembang untuk membangun Interaction Diagram untuk melihat bagaimana objek – objek saling bekerja sama dalam menyediakan fungsionalitas yang diperlukan. Diagram Class juga dapat dibuat untuk melihat kelas – kelas yang terlibat dalam suatu sistem dan bagaimana mereka saling berhubungan satu sama lain. Diagram komponen dapat dikembangkan untuk menggambarkan bagaimana kelas – kelas dipetakan menjadi komponen implementasi. Terakhir deployment diagram dapat juga dikembangkan untuk melihat penyebaran komponen – komponen dalam jaringan komputer yang ada untuk sistem / perangkat lunak yang sedang dikembangkan. Dalam Rational Rose, pemodelan adalah cara melihat sistem dari berbagai sudut pandang. Rational Rose mencakup semua diagram yang dikenal dalam UML, aktor – aktor yang terlibat dalam sistem, use case – use case, objek – objek, kelas – kelas, komponen – komponen *deployment node*. Model juga mendekripsikan rincian yang diperlukan sistem dan bagaimana sistem bekerja, sehingga para pengembang dapat menggunakan model itu sebagai *blue print* untuk sistem yang akan dikembangkan. Beberapa keunggulan Rational Rose dalam pemodelan, diantaranya :

1. Bahasa yang digunakan adalah bahasa pemodelan stándar yaitu UML, dimana notasi stándar dalam UML akan dapat meningkatkan komunikasi antar tim
2. Rational Rose mendukung round trip engineering, sehingga dapat di-*generate* model kedalam kode (Java, C ++, Visual Basic, dan sebagianya) dan melakukan reverse engineering untuk menampilkan arsitektur software dari kode yang ada. Hal ini dapat dilakukan secara bolak – balik sebagai proses interatif selama proses rekayasa software.
3. Model dan kode senantiasa sinkron selama dalam *development cycle*.
4. Membangun software menggunakan Rational Rose memudahkan dalam memperbaiki software tersebut karena apabila suatu saat ditemukan *requirement* baru, dapat kembali digambarkan lagi software tersebut dalam UML.
5. Para user Rational Rose dapat berkomunikasi walaupun bekerja dalam sistem operasi yang berbeda (Windows atau UNIX).
6. Mendukung rekayas software untuk sistem client/server sehingga Rational Rose merupakan software pemodelan visual yang tangguh dalam lingkungan client/server, e-business, dan lingkungan perusahaan terdistribusi (kantor – kantornya terletak dalam tempat terpisah – pisah).

VIEW DALAM RATIONAL ROSE

Setiap sistem yang kompleks selalu akan lebih baik jika didekati melalui himpunan – himpunan sudut pandang yang kecil yang satu sama lain hampir saling bebas. Sudut pandang tunggal senantiasa tidak mencukupi untuk melihat sistem yang besar dan kompleks. Alasan ini menjelaskan mengapa saat dibaca suatu konstruksi suatu gambar sering dilihat berbagai cara pandang (*view*) terhadap gambar tersebut. Demikian pula saat dibuat model untuk membangun suatu software juga terdapat berbagai cara pandang (*view*).

Dalam Rational Rose view tersebut dibagai menjadi :

a. *Use case View*

Use case view membantu untuk memahami dan menggunakan sistem yang dimodelkan. View ini melihat pada bagaimana actor dan use case berinteraksi. Aktor menggambarkan pengguna (user) sistem. Aktor membantu membatasi sistem dan memberi gambaran yang jelas mengenai apa yang harus dilakukan oleh sistem. Penting untuk dicatat bahwa aktor berinteraksi dengan use case tetapi tidak mengendaikan use case. Sebuah use case dapat digambarkan sebagai suatu cara tertentu untuk menggunakan sistem dari sudut pandang pengguna (aktor). Terdapat beberapa diagram yang digunakan dalam use case view, yaitu :

- *Use case Diagram*
- *Sequence Diagram*
- *Collaboration Diagram*
- *Activity Diagram*

b. *Logical View*

Logical view mengarah pada persyaratan (*requirements*) fungsional sistem. View ini melihat kelas – kelas dan hubungan antar kelas – kelas tersebut. Diagram dalam view ini adalah :

- *Class Diagram*
- *Sequence Diagram*
- *Collaboration Diagram*
- *Statechart Diagram*

c. *Component View*

Component view mengarah pada pengaturan software. View ini mengandung informasi mengenai komponen – komponen software, komponen tereksekusi (*executable*) dan libary untuk sistem yang dimodelkan. Hanya ada satu jenis diagram yang digunakan pada view ini, yaitu *component diagram*.

d. *Deployment View*

Deployment view memperlihatkan pemetaan setiap proses ke dalam hardware. View ini paling bermanfaat ketika dibuat model suatu sitem yang diterapkan dalam lingkungan arsitektur yang terdistribusi dimana diterapkan aplikasi dan server pada lokasi yang berbeda.. View ini hanya memiliki satu diagram, yaitu deployment diagram.

DIAGRAM DALAM RATIONAL ROSE

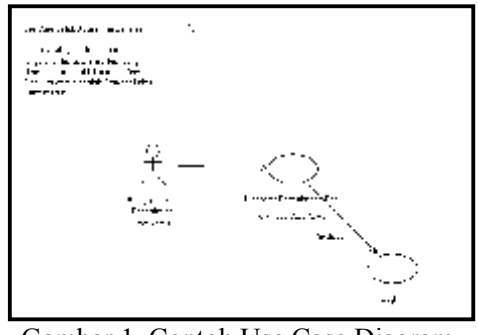
Dalam Rational rose dikenal berbagai macam diagram. Tipe diagram yang berbeda – beda ini membantu untuk melihat sistem dari perspektif yang berbeda – beda.

Semua diagram yang ada dalam UML, terdapat jug dalam Rational Rose. Diagram dalam Rational Rose adalah sebagai berikut :

1. Diagram Use Case

Diagram use case menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (actor). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar.

Diagram use case dapat digunakan selama proses analisis untuk menangkap requirements sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, use case diagram menetapkan perilaku (*behavior*) sistem saat diimplementasikan.



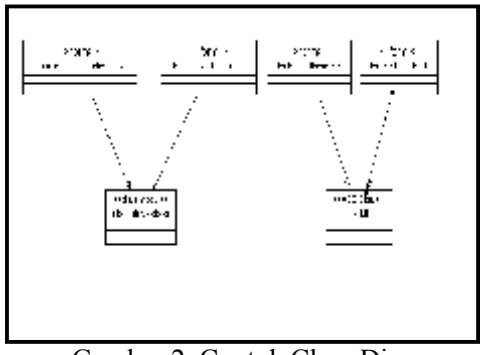
Gambar 1. Contoh Use Case Diagram

2. *Class Diagram*

Class diagram membantu dalam visualisasi struktur kelas – kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak. Class diagram memperlihatkan hubungan antar kelas dan penjelasan detail tiap – tiap kelas di dalam model desain (dalam logical view) dari suatu sistem. Selama proses analisi, class diagram memperlihatkan aturan – aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Selama proses analisis, *class diagram* memperlihatkan aturan – aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap decían, class diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

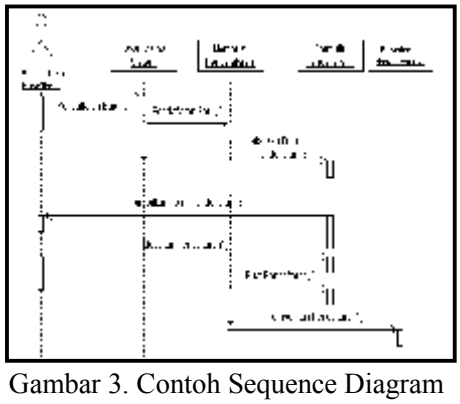
Class diagram juga merupakan fondasi untuk *component diagram* dan *deployment diagram*.



Gambar 2. Contoh Class Diagram

3. Sequence Diagram

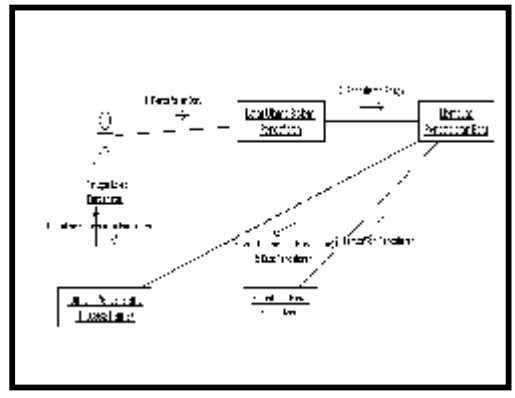
Diagram sequence menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosiasi dengan use case. Sequence diagram memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu di dalam use case. Diagram sequen sebaiknya digunakan diawal tahap desain atau analisis karena kesederhanaannya dan mudah untuk dimengerti.



Gambar 3. Contoh Sequence Diagram

4. Collaboration Diagram

Diagram kolaborasi memperlihatkan interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (relationship) antar objek, sedangkan sequence diagram menekankan pada urutan kejadian. Dalam satu diagram kolaborasi terdapat beberapa object, link, dan message. Diagram kolaborasi digunakan sebagai alat untuk menggambarkan interaksi yang mengungkapkan keputusan mengenai perilaku sistem.

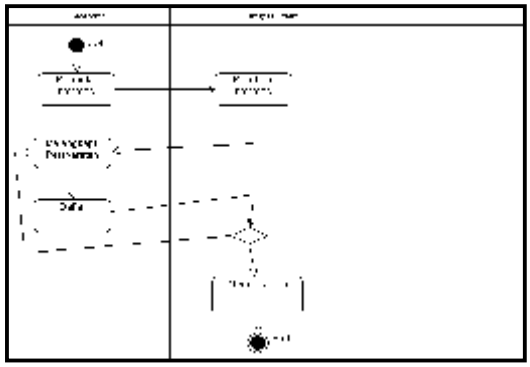


Gambar 4. Contoh Collaboration Diagram

5. Activity Diagram

Activity diagram memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah flowchart karena dapat dimodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas ke dalam keadaan sesaat (*state*). Seringkali bermanfaat bila dibuat sebuah activity terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. Activity diagram juga sangat berguna ketika ingin menggambarkan perilaku paralel atau menjelaskan bagaimana perilaku dalam berbagai use case berinteraksi. Dapat digunakan statechart diagram untuk memodelkan perilaku dinamis satu kelas atau objek. Statechart diagram memperlihatkan urutan keadaan sesaat (*state*) yang dilalui sebuah objek, kejadian yang menyebabkan sebuah transisi dari satu state atau aktivitas ke state atau aktivitas lainnya, dan aksi yang menyebabkan perubahan satu state lainnya, dan aksi yang menyebabkan perubahan satu state atau aktivitas. Diagram aktivitas paling cocok

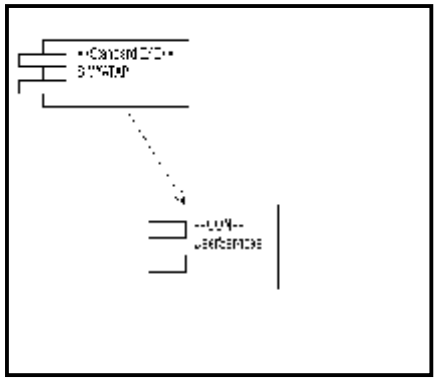
digunakan untuk memodelkan urutan aktivitas dalam suatu proses.



Gambar 5. Contoh Activity Diagram

6. *Component Diagram*

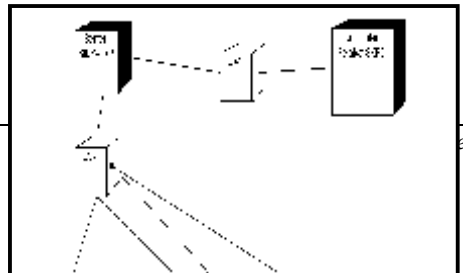
Component diagram menggambarkan alokasi semua kelas dan objek kedalam komponen – komponen dalam desain fisik sistem software. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen – komponen software, seperti source code, binary code, dan komponen tereksekusi (*executable components*). Dapat dibuat satu atau lebih *component diagram* untuk menggambarkan komponen dan paket atau menerangkan isi dari tiap – tiap paket komponen.



Gambar 6. Contoh *Component Diagram*

7. Deployment Diagram

Setiap model hanya memiliki satu diagram deployment. Diagram ini memperlihatkan pemetaan software ke hardware.



Gambar 7. Contoh Deployment Diagram

KESIMPULAN

Pemodelan visual (visual modeling) merupakan proses menggambarkan cetak biru suatu sistem informasi secara grafis, terdiri dari komponan – komponen, interface, dan koneksi – koneksi yang ada dalam sistem tersebut, agar mudah dipahami dan dikomunikasikan. Visual modeling dapat membantu untuk menampilkan elemen – elemen yang penting secara detail dari suatu masalah yang kompleks dan menyaring untuk kemudian membuang elemen – elemen yang tidak penting.

Dengan menggunakan model diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti scalability, robustnees, security, dan sebagainya. Untuk melakukan pemodelan sistem / perangkat lunak secara visual digunakan UML (Unified Modelling Language) yang digambarkan secara elektronik lewat sarana perangkat lunak Rational Rose.

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Rational Rose merupakan *tool* pemodelan secara visual untuk pengembangan sistem berbasis objek yang sangat handal untuk digunakan sebagai bantuan bagi para pengembang dalam melakukan analisis dan perancangan sistem.

DAFTAR PUSTAKA

1. Nugroho, A., (2005), "*Analisis dan Perancangan Sistem Informasi Dengan Metodologi Berorientasi Objek"*, Penerbit Informatka, Bandung.
2. Nugroho, A., (2005), "*Rational Rose untuk Pemodelan Berorientasi Objek*", Penerbit Informatika, Bandung.
3. Hariman, A.S., (2002), "*Visual Modelling menggunakan UML dan Rational Rose"*, Penerbit Informatika Bandung.